Fast Attack Recovery for Stochastic Cyber-Physical Systems

Lin Zhang*
University of Pennsylvania
cpsec@seas.upenn.edu

Luis Burbano*
University of California, Santa Cruz
lburbano@ucsc.edu

Xin Chen
University of New Mexico
chenxin@unm.edu

Alvaro A. Cardenas
University of California, Santa Cruz
alacarde@ucsc.edu

Steven Drager, Matthew Anderson
Air Force Research Laboratory
{steven.drager, matthew.anderson.37}@us.af.mil

Fanxin Kong
University of Notre Dame
fkong@nd.edu

***Abstract*—Cyber-physical systems tightly integrate computational resources with physical processes through sensing and actuating, widely penetrating various safety-critical domains, such as autonomous driving, medical monitoring, and industrial control. Unfortunately, they are susceptible to assorted attacks that can result in injuries or physical damage soon after the system is compromised. Consequently, we require mechanisms that swiftly recover their physical states, redirecting a compromised system to desired states to mitigate hazardous situations that can result from attacks. However, existing recovery studies have overlooked stochastic uncertainties that can be unbounded, making a recovery infeasible or invalidating safety and real-time guarantees. This paper presents a novel recovery approach that achieves the highest probability of steering the physical states of systems with stochastic uncertainties to a target set rapidly or within a given time. Further, we prove that our method is sound, complete, fast, and has low computational complexity if the target set can be expressed as a strip. Finally, we demonstrate the practicality of our solution through the implementation in multiple use cases encompassing both linear and nonlinear dynamics, including robotic vehicles, drones, and vehicles in high-fidelity simulators.**

***Index Terms*—cyber-physical systems, security, sensor attack, recovery, real-time, stochastic systems**

I. Introduction

Cyber-Physical Systems (CPS) integrate computation, networking, and physical processes, where embedded computers and networks monitor and control the physical processes via sensing and actuation. The functionality and connectivity in CPS continue to increase and enable advanced applications such as autonomous and connected vehicles, unmanned aerial vehicles, and smart grids. Despite several advantages, CPSs are vulnerable to various attacks, such as physical interference affecting sensor readings [1] and actuation signals [2], GPS spoofing [3], etc. Such security lapses in CPS can cause dangerous consequences in the physical world, such as power outages [4], car accidents [5], navigation errors [6], or damage to nuclear power systems [7], [8].

To protect these systems, researchers have developed several tools for preventing, detecting, and recovering from attacks. Real-time attack recovery solutions can be broadly classified into three categories: (1) restarting a system [9]–[11], (2) replacing compromised sensor data with predicted one from virtual sensors [12]–[14], and (3) replacing the original control algorithm by a backup controller [15]–[17]. The first two categories do not provide strong safety or timing guarantees; however, they are simple solutions that can be easy to implement and offer minimal computational overhead. The last category requires us to design a new safety controller to take over a system under attack. A provably safe controller can provide strong safety and timing guarantees, but this design can increase the computational complexity of the recovery problem. Moreover, solutions with guarantees are an all-or-nothing approach [18], [19] (they either guarantee the safety and real-time requirements or give up searching for a feasible solution). Further, to ensure safety and real-time guarantees, they assume bounded uncertainties, which does not hold in real applications since most real-world CPSs are riddled with stochastic uncertainties, including modeling errors, external disturbances, and sensor noise. Furthermore, these bounded uncertainties can result in a conservative recovery deadline, exacerbating the infeasibility of deterministic approaches.

In this paper, we explore "Can we design an efficient recovery controller that always finds a solution (completeness) for stochastic CPS and offers provable safety and timing properties (soundness)?" Instead of trying to find an all-or-nothing approach, we focus on finding a solution that maximizes the probability of safe recovery within a given time. In particular, this paper proposes the *optimal probabilistic recovery (OPR)* problem, i.e., designing a controller that takes over the stochastic CPS after an attack and steers the system to reach a *target set* as fast as possible and with the highest probability of reaching the desired set. Then, we put forward an efficient algorithm to solve this problem and prove that this method is sound when the target set can be defined as a strip. In this way, we pay attention to the recovery probability instead of the deterministic guarantees, making the recovery complete. In addition, the evaluations show that our recovery strategy is efficient and still performs well and outperforms previous works even when regularity assumptions do not hold (e.g., drones or robotic vehicles).

Our contributions include the following: (1) We propose a formal framework to design a safe attack-recovery controller for stochastic systems. (2) We prove the soundness and completeness of our solution, and show that our solution

Assigned CLEARED 21 December 2023, case number AFRL-2023-5819.
* These authors contributed equally.

maximizes the probability of the fastest recovery with safety guarantees. (3) We show the practicality of our solution in various cases, ranging from high-fidelity models such as SVL and AirSim/Gazebo, real-world robotic vehicles, and several simulations for linear systems. (4) We compare our solution with previous proposals and illustrate the benefits of our solution under key performance metrics.

The remainder of the paper is organized as follows. Section II discusses related work and highlights our contributions. Section III provides system and threat models and problem statement. Section IV introduces our attack-recovery method. Section V evaluates our proposed method and compares with existing proposals. Section VI concludes the paper.

TABLE I: Classification of work on online attack recovery

	[12]	[38]	[14]	[15]	[19]	[35]	Ours
Guarantees							
Safety	-	-	-	●	●	-	●
Real-time	-	-	-	●	●	-	●
Formal Properties							
Soundness	-	●	-	●	●	-	●
Completeness	●	●	●	-	-	●	●
System Type							
Non-linear	◐	-	◐	-	●	●	●
Stochastic	-	●	-	-	-	-	●

Legend: ●: feature considered by authors. ◐: feature considered for results but not for the design. -: feature not considered by authors.

II. Related Work

There is a growing literature on the security of CPS. One of the research areas that has attracted great interest is CPS attack detection [20]–[24]. Most of these efforts focus on detecting attacks, leaving the response to intrusions as future work: for example, a survey paper in 2021 on drone security summarizes several attack prevention and detection mechanisms; however, only two out of 131 citations focus on attack response (adding parachutes to drones) [25].

Automated attack response is a growing area in classical computer networks to mitigate the time-consuming and delayed response of overwhelmed security analysts. Automated actions include patching, restoring software, blocking an attacker's IP addresses, switching to another server, or restarting systems [26], [27]. However, these automated attack responses for computer networks are not adequate for CPS. Since the CPS actively interacts with the physical world, a malicious sensor attack can lead to severe consequences, such as personal casualties and economic losses.

Researchers have been working on new online attack recovery strategies to mitigate the physical impact of attacks. These strategies can be broadly classified into three categories: *i) Restart-Based Recovery (RR)*. To respond to a detected attack, RR restarts a CPS [9]–[11]. This strategy has been mainly used for attacks on the controller [28], but sensor attacks may continue to affect systems after restart. *ii) Virtual Sensors (VS)*. VS approaches replace corrupted sensor data with predicted data by a software-simulated sensor and rely on the nominal controller to drive the system to the desired physical state [12], [14], [29], [30]. *iii) Backup Controller (BC)*. This recovery strategy extends the simplex architecture [16], [31], [32] and robust control [33], [34]. It requires designing a backup controller that takes over the system to recover it after an attack is detected [15], [18], [35]–[37].

For a deeper understanding of the online attack recovery for CPS, we summarize the features of these methods in terms of recovery goals, formal properties, and target system type.

Recovery Goals: We identify that researchers design the above strategies to preserve two properties of attack recovery: *safety*, and *real-time*. *i) Safety.* The goal of strategies focused on safety is to protect humans and CPS from dangerous conditions, i.e., away from unsafe states during recovery [15], [18], [19]. For example, we want a vehicle under GPS spoofing attack to be steered to a safe zone, such as a roadside shoulder, to avoid further deviation or collision and guarantee safety. If there is no safety consideration or guarantee, the recovery is as damaging as attacks since severe consequences, such as collisions, may occur during recovery. For this reason, the paper aims to preserve safety under attack instead of just fulfilling its original mission. *ii) Real-time.* Timing is vital for safety-critical CPS, and untimely recovery can lead to severe consequences. Thus, some research starts to consider recovery time, i.e., the interval between the attack detection and the end of recovery. Some real-time recovery methods online estimate a conservative deadline to avoid unsafe physical states and complete the recovery before this deadline [15], [18], [19]. However, finding a suitable deadline is time-consuming, and sometimes the deadline is too conservative to find a feasible recovery solution. Thus, this paper aims to recover the physical system fastest rather than estimating a deadline first.

Formal Properties: When dealing with adversarial conditions, recovery methods should have some formal properties. We distinguish between soundness and completeness as two formal properties: *i) Soundness* means that once we find a recovery control sequence, it guarantees that the system can be driven back to the desired states under safety and timing restrictions. *ii) Completeness* means that the recovery strategies can always find a recovery control sequence. Soundness can be achieved by using reachability analysis to obtain a recovery deadline and then formulate an optimization problem to produce a control sequence with a length equal to the deadline [15], [18]. However, these methods are not complete. Reachability analysis overapproximates the reachable set given an uncertainty bound, leading to an unnecessarily conservative recovery deadline. The deadline can result in infeasible optimization problems, i.e., failing to find a recovery control sequence.

System Type: To reduce computational overhead, several authors consider systems with *linear* dynamics to synthesize recovery strategies [14], [38]. However, the linear dynamics are not enough to capture the richer behavior of systems, since real systems such as autonomous vehicles are typically nonlinear. If we apply these strategies directly to real systems, they may fail to recover systems. Therefore, we require recovery

strategies that can handle more complex systems that present *nonlinear* dynamics. Moreover, some recovery strategies consider uncertainties inherently in real systems, such as sensor noise. Analyzing such *stochastic systems* has been a significant research topic in formal methods and control theory [39]–[43]. Most existing techniques find control inputs (e.g., the vehicle steering) by solving an optimization problem that describes the goal and a discrete abstraction of the stochastic dynamics [44]–[46]. Due to the high computational complexities, they are mostly developed for offline use and cannot be directly used to perform online recovery for stochastic systems.

Difference from Existing Work. Table I summarizes and categorizes the most related work on online attack recovery among all the features above. Compared to existing methods, this paper focuses on *online, autonomous* attack recovery mechanisms for CPS. In particular, it develops a BC that prioritizes the safety of humans and system and redirects the physical system to a target set of desired states fastest within a given time to prevent further damage caused by the attack. Moreover, the strategy is proven to be sound and complete when the target set is defined as a strip. Further, it can effectively and efficiently handle linear and nonlinear systems with stochastic uncertainties.

III. Preliminaries

A. System Model

We consider a CPS consisting of a physical plant, nominal controllers, actuators, and sensors. The physical plant follows physical laws and can be expressed as a continuous model using a differential equation $\dot{\boldsymbol{x}} = f_c(\boldsymbol{x}, \boldsymbol{u})$, where $\boldsymbol{x} \in \mathbb{R}^n$ is physical states (such as temperature and velocity), $\dot{\boldsymbol{x}}$ is the derivative of $\boldsymbol{x}$, and $\boldsymbol{u} \in \mathbb{R}^m$ is control input (such as voltage and throttle) that drives actuators (such as the motor of a vehicle) to modify the physical state $\boldsymbol{x}$ and completing the CPS task. The model, typically nonlinear in real systems, reflects how system states evolve with control inputs. The system states typically follow reference (or target) states to maintain desired behaviors. For instance, cruise control requires that the car maintain a steady speed as set by the driver. To achieve this, the nominal controller periodically generates a piecewise constant control input signal. At each control step, it estimates system states $\boldsymbol{x}_t$ at time t based on sensor measurements $\boldsymbol{y}_t$ and then generates a control input $\boldsymbol{u}_t$ sent to actuators. The control input signal is limited by the physical properties of actuators, so the control inputs used at each time are bounded in the set $\mathcal{U} \subset \mathbb{R}^m$. Since the control input signal is discrete, the system model can also be expressed in a discrete form

$$\boldsymbol{x}_{t+1} = f_d(\boldsymbol{x}_t, \boldsymbol{u}_t) + C\boldsymbol{w}_t, \tag{1}$$

where $\boldsymbol{w}$ is a vector representing the uncertainty in the system (a stochastic variable). The system state at a time is subject to a probabilistic distribution, which is dependent on the initial state and the randomness in the current and past steps. We may also associate a probabilistic distribution on the initial state $\boldsymbol{x}_0$. Because of the central limit theorem, we consider the uncertainty of the initial state and the noise $\boldsymbol{w}$ to be distributed as the standard Gaussian distribution $\mathcal{N}(0, 1)$ with covariance matrix C. Given an initial state $\boldsymbol{x}_0$, any state $\boldsymbol{x}_t$ for $t = 0, 1, \ldots$ is called a *reachable state* from $\boldsymbol{x}_0$. Stochastic dynamics consider uncertainties with a known probability distribution that affects the evolution and observation of the state variables. Such uncertainties are inherent in real-world CPS and could come from measurement noises, external disturbances, etc. In contrast, models with deterministic or bounded nondeterministic dynamics do not reflect such stochastic behaviors of systems, so they are limited when used in real-world CPS.

B. Threat Model and Assumptions

We present our recovery approach from the perspective of sensor attacks that become a crucial threat targeting CPS. The adversaries can manipulate sensor measurements by compromising the integrity of information perceived by the CPS (e.g., spoofing and transduction attacks) or the availability of information (e.g., DoS attacks). Specifically, the adversaries change the actual sensor data y_t at time t to $\hat{y}_t$ observed by the nominal controller, where $y_t \neq \hat{y}_t$. As a result, the controller generates a corrupt control input, driving the physical system towards unsafe states. Note that we make no assumptions about the number of compromised sensors; it could be one or many (we discuss how to deal with the number of compromised sensors in Section IV-D).

To bypass security countermeasures, attackers are incentivized to launch non-invasive sensor attacks, such as transduction attacks, which leave actuators and controllers intact. Thus, we assume that the actuators must be able to implement commands from the recovery controller correctly and that the recovery controller must remain uncompromised to take control after the attack has been detected. Further, recovery against other attacks, such as actuator and controller attacks, which require additional adaptation, is beyond the scope.

Since this paper focuses on recovery, we also assume that an attack detection system is in place, such as [47]–[50]. During this detection delay between the onset of an attack and its detection, the controller will operate with the corrupted values, which may force the physical system from a desired trajectory toward an unsafe condition.

C. Problem Statement

We consider a nonlinear CPS with stochastic uncertainties expressed in Section III-A suffering from attacks described in Section III-B. Fig. 1 demonstrates a timeline of the recovery process we want to achieve for a CPS under sensor attack. The system initially runs a nominal controller. The adversary launches a sensor attack starting at time t_a, and the system states start to deviate from the desired states. Once the attack detector identifies the attack at time t_r, the system switches to a recovery controller. The safety controller takes over the entire system and generates a recovery control sequence to steer the physical state to the target set as quickly as possible before a given time. This sequence must maximize the probability that the final state x_e is within the target set.

For example, in Fig. 2, autonomous vehicles follow their appropriate lane on a congested road. An attack on the

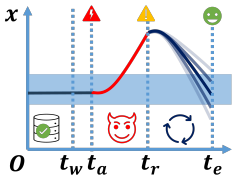


Fig. 1: Timeline.

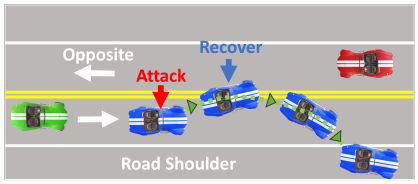


Fig. 2: Recovery Use-Case.

perception system of the blue car causes it to swerve into oncoming traffic. Once an attack is detected, we need to attempt to steer the car to a safer region (target set) within several seconds to avoid collision with an ongoing red car. We can choose different target set options depending on the road and traffic conditions: one option is to return the car to the original lane; if there is a road shoulder, a better option would be to steer the car out of the highway and pull it over to the shoulder to avoid a rear collision with the green car.

D. Recovery Goals

An ideal solution to this problem should satisfy the following objectives. **(i) Safety:** an attack-recovery solution should steer a CPS to a target state set where it cannot harm humans or other systems. The target set should be determined based on domain knowledge and customized to the particular application [51]. For example, we can attempt to guide the car to the road shoulder in the aforementioned example. **(ii) Robustness to Uncertainty:** an attack-recovery algorithm should be robust to stochastic uncertainty in its environment. **(iii) Fast Recovery:** we want a solution that drives the CPS as fast as possible to the target set to prevent operational errors from building up and causing accidents. **(iv) Low Computational Overhead:** because the software of CPS runs on embedded systems, we also want a recovery solution with minimal computational overhead. **(v) Soundness:** the safety controller should only give control actions that satisfy our desired properties. **(vi) Completeness:** the safety controller always finds a solution that meets our objectives.

IV. Real-Time Attack Recovery

This section first sketches the online attack recovery process. Then, it presents the core component, Optimal Probabilistic Recovery (OPR), and shows how it achieves the properties mentioned above. In addition, it describes other components and explains how they support OPR.

A. Online Recovery Overview

The online attack recovery structure is illustrated in Fig. 3. Once an attack is detected, the system switches from nominal mode to recovery mode, where the online recovery controller (blue-shaded dashed box) takes over the system. The core component *Optimal Probabilistic Recovery (OPR)* generates a recovery control sequence that drives the physical states to the target set fastest with the highest probability of being in this set within a given time (Section IV-B and IV-C). Meanwhile, the *State Reconstruction* rebuilds the corrupted state estimate and provides the initial state probabilistic distribution for

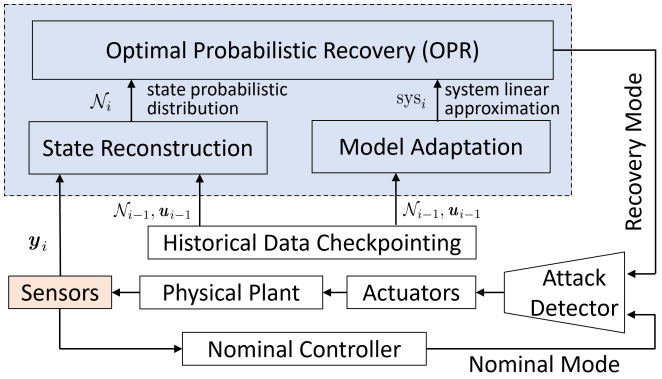


Fig. 3: Overview of Our Attack Recovery Structure.

OPR; the *Model Adaption* keeps updating the system linear approximation (Section IV-D).

Algorithm 1 Online Predictive Recovery

Input: $\boldsymbol{x}_w, \boldsymbol{u}_w, \ldots, \boldsymbol{u}_{r-1}, e_{max}$.
Output: $\boldsymbol{u}_r, \ldots, \boldsymbol{u}_{e-1}$ # Recovery control sequence

1: **for** $i \leftarrow r$ to e_{max} **do**
2: **if** $i == r$ **then**
3: $\mathcal{N}_i \leftarrow \text{StateReconstruction}(\boldsymbol{x}_w, \boldsymbol{u}_w, \ldots, \boldsymbol{u}_{r-1})$;
4: **else**
5: $\mathcal{N}_i \leftarrow \text{StateReconstruction}(\mathcal{N}_{i-1}, \boldsymbol{u}_{i-1})$;
6: **end if**
7: $\text{sys}_i \leftarrow \text{ModelAdaptation}(\mathcal{N}_i, \boldsymbol{u}_{i-1})$;
8: $\boldsymbol{u}'_0, \ldots, \boldsymbol{u}'_k \leftarrow \text{OPR}(\text{sys}_i, \mathcal{N}_i)$; # Section IV-B, IV-C
9: $\boldsymbol{u}_i \leftarrow \boldsymbol{u}'_0$ # Only apply the first control input
10: **if** $k == 0$ **then**
11: $e \leftarrow i$; # The physical state has the highest
12: break; # probability of being in the target set
13: **end if**
14: **end for**

Algorithm 1 describes how these three components cooperate with each other. At time t_r, the system notices that an attack has compromised the sensor data before the attack is detected, and the state estimate cannot reflect the actual state. Thus, on Line 3 of the algorithm, *State Reconstruction* first rebuilds the current state probabilistic distribution (PD) $\mathcal{N}_r$ from checkpointed trustworthy data. Moreover, the physical states have been driven away from the desired states. Thus, from t_r, the recovery mode kicks in to generate a recovery control sequence until it either drives the physical state to the target set with the highest probability (Lines 10-12) or runs out of the given time e_{max} (Line 1): At the i^{th} control step, *State Reconstruction* first estimates the current state PD based on the last PD and the last recovery control input (Line 5). Then, *Model Adaption* on Line 7 obtains a Linear Time-Invariant (LTI) model from nonlinear dynamics around the current state and control input. Based on them, *OPR* performs an efficient algorithm to compute recovery control inputs (Line 8). Note that it only applies the first recovery input to the actuator at each control step (Line 9).

B. Optimal Probabilistic Recovery Problem

We first formally define the OPR problem as Definition IV.1.

Definition IV.1 (Optimal Probabilistic Recovery (OPR) Problem)**.** *Given a stochastic system in the form of Eq. (1), a target set $\mathcal{T}$, and an integer $K > 0$, can we find a control sequence $\hat{\boldsymbol{u}}_0, \ldots, \hat{\boldsymbol{u}}_{k-1}$ for some $1 \le k \le K$ such that it has the highest probability of steering the system to the target from all control sequences in less than K control steps?*

Given the range $\mathcal{U}$ for all control inputs, if we denote the probability of reaching the target set at step k via the control sequence $\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{k-1} \in \mathcal{U}$ by $\mathcal{P}(\boldsymbol{x}_k(\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{k-1}) \in \mathcal{T})$, then the OPR solves the following optimization problem

$$\begin{aligned} \max_{1\le k\le K, \boldsymbol{u}_0,\ldots,\boldsymbol{u}_{k-1}\in\mathcal{U}} \; & \mathcal{P}(\boldsymbol{x}_i(\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{i-1}) \in \mathcal{T}) \\ \text{s.t.} \quad & \boldsymbol{u}_0,\ldots,\boldsymbol{u}_{i-1} \in \mathcal{U}, i \le K. \end{aligned} \tag{2}$$

Finding optimal controllers for discrete-time stochastic dynamics has been intensively studied in recent years [46], [52], [53]. Most approaches are developed based on reachability computation [54]–[56]. However, they are used for more general purposes, such as solving optimal reach-avoid control problems (e.g., reaching a destination while avoiding obstacles). Thus, most of them require performing state-space abstraction and/or solving nonlinear constraints, making them too costly to be used online and unsuitable for real-time usage.

Fortunately, we find that most target sets in CPS can be defined as a strip, i.e., the region between two parallel hyperplanes: $\mathcal{S} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \ell^T \boldsymbol{x} \ge c_1 \text{ and } \ell^T \boldsymbol{x} \le c_2\}$, where $c_1, c_2 \in \mathbb{R}$, $c_1 \le c_2$, and $\ell \in \mathbb{R}^n$ is a normal vector of two hyperplanes. For example, we can express that the temperature of a reactor should be between $350^\circ F$ and $360^\circ F$ as a strip.

Once the target set can be defined as a strip, we propose a new approach in the next subsection to solve the OPR problem on LTI models without using any linear or nonlinear optimizer. We further show that our method is not only a sound and complete solution but also has low computational complexity.

C. Algorithm for Solving OPR Problem

Given the initial PD from *State Reconstruction* and updated LTI approximation from *Model Adaptation*, Fig. 4 shows that the algorithm performs a reachability search on the mean of reachable PD and obtains a control sequence that maximizes the probability of successful recovery within K steps.

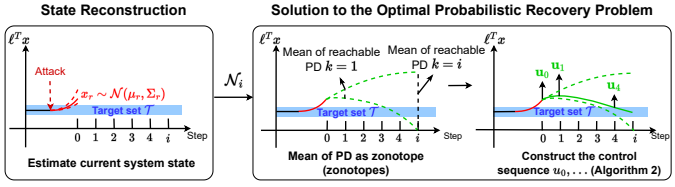


Fig. 4: Illustration of solving the OPR problem.

1) Reachable PD Analysis: Before we introduce the main algorithm, let us investigate the stochastic behavior of the LTI approximation at each time step

$$\boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t + C\boldsymbol{w}_t \tag{3}$$

Given that the initial state $\boldsymbol{x}_0$ is subject to a Gaussian distribution $\mathcal{N}(\mu_0, \Sigma_0)$, then the state at the next step $\boldsymbol{x}_1 = A\boldsymbol{x}_0 + B\boldsymbol{u}_0$ via a control input $\boldsymbol{u}_0$ is still subject to a Gaussian distribution that is defined by $\mathcal{N}(A\mu_0 + B\boldsymbol{u}_0, A\Sigma_0 A^T + CC^T)$, since the independent multivariate Gaussian distributions are closed under additions and linear transformations. I.e., given $x \sim \mathcal{N}(\mu_x, \Sigma_x)$, $y \sim \mathcal{N}(\mu_y, \Sigma_y)$, we have that

$$x + y \sim (\mu_x + \mu_y, \Sigma_x + \Sigma_y), \quad Mx \sim (M\mu_x, M\Sigma_x M^T).$$

Therefore, given a sequence of k control inputs $\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{k-1}$, the reachable state $\boldsymbol{x}_k(\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{k-1})$ can be expressed as

$$\boldsymbol{x}_k = A^k \boldsymbol{x}_0 + \underbrace{\sum_{i=1}^{k} A^{i-1} B \boldsymbol{u}_{k-i}}_{\overline{u}_k} + \underbrace{\sum_{i=1}^{k} A^{i-1} C \boldsymbol{w}_{k-i}}_{\overline{w}_k} \tag{4}$$

and $\boldsymbol{x}_k$ is subject to the Gaussian distribution

$$\boldsymbol{x}_k \;\sim\; \mathcal{N}(A^k\mu_0 + \overline{\boldsymbol{u}}_k\,,\, A^k\Sigma_0(A^k)^T + \Sigma_{\overline{w}_k}) \tag{5}$$

wherein $\Sigma_{\overline{w}_k} = \sum_{i=1}^{k} A^{i-1} CC^T (A^{i-1})^T$. Additionally, the term $\overline{w}_k$ representing the accumulation of noises is also subject to a Gaussian distribution, which is $\mathcal{N}(0, \Sigma_{\overline{w}_k})$.

PD Decomposition. To find a $\boldsymbol{x}_k$ with the highest probability of staying in the target set, we need to find the optimal PD under all possible control sequences. To do so, we investigate the probability of a random vector staying in a strip. A PD can be decomposed into mean and covariance. Lemma IV.1 tells that given two Gaussian distributions of the same covariance, the one with a closer mean to the strip center has a higher probability of staying in the strip. Since the covariance of the distribution (5) is identical at the k^{th} step, we may design an algorithm to *find the closest mean* to the center of the strip among all possible control sequences bounded by a user-given maximum length K.

Lemma IV.1. *Given a strip $\mathcal{T}(\boldsymbol{x}) : a \le \ell^T \boldsymbol{x} \le b$, and two Gaussian distributions $\boldsymbol{y}_1 \sim \mathcal{N}(\mu_1, \Sigma)$ and $\boldsymbol{y}_2 \sim \mathcal{N}(\mu_2, \Sigma)$ which have the same covariance matrix. We have that $\mathcal{P}(\boldsymbol{y}_1 \in \mathcal{T}(\boldsymbol{y}_1)) \le \mathcal{P}(\boldsymbol{y}_2 \in \mathcal{T}(\boldsymbol{y}_2))$ iff. $|\frac{a+b}{2} - \ell^T\mu_1| \ge |\frac{a+b}{2} - \ell^T\mu_2|$.*

Proof. The probability of $\mathcal{P}(\boldsymbol{x} \in \mathcal{T}(\boldsymbol{x}))$ w.r.t. a Gaussian distribution $\boldsymbol{x} \sim \mathcal{N}(\mu, \Sigma)$ can be computed accurately by first linear transforming the distribution of $\boldsymbol{x}$ to a univariate Gaussian distribution $z \sim \mathcal{N}(\ell^T\mu, \ell^T\Sigma\ell)$, and then evaluate

$$\begin{aligned} \mathcal{P}(a \le z \le b) &= \frac{1}{2}\left(\text{erf}(\frac{b-\ell^T\mu}{\sqrt{2\ell^T\Sigma\ell}}) - \text{erf}(\frac{a-\ell^T\mu}{\sqrt{2\ell^T\Sigma\ell}})\right) \\ &= \frac{1}{\sqrt{\pi}} \int_{\frac{a-\ell^T\mu}{\sqrt{2\ell^T\Sigma\ell}}}^{\frac{b-\ell^T\mu}{\sqrt{2\ell^T\Sigma\ell}}} e^{-t^2} dt. \end{aligned}$$

When a, b, ℓ, Σ are constant, the above expression monotonically increases along with the decreasing of $|\frac{a+b}{2} - \ell^T\mu|$. To

see this fact, we introduce $\Delta = \frac{a+b}{2} - \ell^T\mu$ to replace $\ell^T\mu$ in the above expression, and computed its derivative w.r.t. Δ:

$$\frac{d\mathcal{P}(a \le z \le b)}{d\Delta} = \frac{1}{\sqrt{2\pi\ell^T\Sigma\ell}}\left(e^{-\frac{(\Delta+\frac{b-a}{2})^2}{2\ell^T\Sigma\ell}} - e^{-\frac{(\Delta-\frac{b-a}{2})^2}{2\ell^T\Sigma\ell}}\right)$$

When $\Delta = 0$, the derivative is 0 and $\mathcal{P}(a \le z \le b)$ has its highest value. When $\Delta < 0$, we have that $(\Delta + \frac{b-a}{2})^2 < (\Delta - \frac{b-a}{2})^2$, the derivative is positive and $\mathcal{P}(a \le z \le b)$ monotonically increases when $\Delta \to 0^-$. On the other hand, when $\Delta > 0$, $(\Delta + \frac{b-a}{2})^2 > (\Delta - \frac{b-a}{2})^2$, the derivative is negative, and then $\mathcal{P}(a \le z \le b)$ monotonically decreases when $\Delta \to +\infty$. Besides, we also have that $\mathcal{P}(\boldsymbol{y}_1 \in \mathcal{T}(\boldsymbol{y}_1)) = \mathcal{P}(\boldsymbol{y}_2 \in \mathcal{T}(\boldsymbol{y}_2))$ if $|\frac{a+b}{2} - \ell^T\mu_1| = |\frac{a+b}{2} - \ell^T\mu_2|$. □

Express Mean of PD as Zonotope. All control inputs are in $\mathcal{U}$. $\mathcal{U}$ can be derived from the physical limitation of a system and is assumed to be a hyperrectangle (box). Then the following set defines all possible means of the distribution (5):

$$\begin{aligned} M_k =& \{A^k\mu_0 + \sum_{i=1}^{k} A^{i-1}B\boldsymbol{u}_{k-i} \mid \boldsymbol{u}_0, \dots, \boldsymbol{u}_{k-1} \in \mathcal{U}\} \\ =& A^k\{\mu_0\} \oplus \bigoplus_{i=1}^{k} A^{i-1}B\mathcal{U} \end{aligned}$$

wherein $\oplus$ denotes the Minkowski sum such that $X \oplus Y = \{x + y \mid x \in X, y \in Y\}$. If we can compute this set for all $k = 1, \dots, K$, then the optimal mean can be obtained by finding the closest point in those sets to the strip center.

Since $\mathcal{U}$ is a box and μ_0 is a point, the set M_k can be viewed as the Minkowski sum of points and linearly transformed boxes, i.e., it is a *zonotope* [57]. Zonotopes are centrally symmetric and closed under linear transformation and Minkowski sum, i.e., the linearly transformed boxes are zonotopes, and the Minkowski sum of zonotopes is still a zonotope. Hence, we seek to compute the set M_k as a zonotope.

$\mathcal{G}$-Representation of a Zonotope. Given a zonotope Z defined by the mapping $\boldsymbol{x} \mapsto G\boldsymbol{x} + c$ from the p-dimensional unit box $\mathbb{B}_p$. It can be represented by its center c along with the generators $g_1, \dots, g_p$ which are the columns of G: $Z = (c, \langle g_1, \dots, g_p\rangle)$. On the other hand, it can also be viewed as the Minkowski sum of the center c and the line segments $L_i = \{\alpha_i g_i \mid \alpha_i \in [-1, 1]\}$ for all $i = 1, \dots, p$:

$$Z = \{c + \sum_{i=1}^{p} a_i g_i \mid a_i \in [-1, 1]\}.$$

When zonotopes are represented in their $\mathcal{G}$-representations, their linear transformation and Minkowski sums can be computed easily. Given $Z = (c, \langle g_1, \dots, g_p\rangle)$, its linear transformation is $MZ = (Mc, \langle Mg_1, \dots, Mg_p\rangle)$. Given two zonotopes $Z_1 = (c_1, \langle g_1, \dots, g_p\rangle)$ and $Z_2 = (c_2, \langle h_1, \dots, h_q\rangle)$, their Minkowski sum is $(c_1 + c_2, \langle g_1, \dots, g_p, h_1, \dots, h_q\rangle)$. Then, if the box $\mathcal{U}$ is represented as $(c_u, \langle g_1^u, \dots, g_m^u\rangle)$ where c_u is the center of $\mathcal{U}$, g_j^u is an m-dimensional column vector whose jth component is the radius of $\mathcal{U}$ in the jth dimension and the others are 0. Intuitively, g_j^u denotes the radius of the jth input range. Then the $\mathcal{G}$-representation of M_k is

$$\begin{aligned} M_k = (&A^k\mu_0 + \sum_{i=1}^{k} A^{i-1}Bc_u, \\ &\langle Bg_1^u, \dots, Bg_m^u, \dots, A^{k-1}Bg_1^u, \dots, A^{k-1}Bg_m^u\rangle), \end{aligned}$$

which has km generators each of which is n-dimensional.

Zonotopes in Reachability Analysis. Zonotopes have been widely used as a reachable set representation for linear and nonlinear systems [58], [59]. They are even effectively used in online predictive monitoring [60] and provide probabilistic hulls for linear stochastic systems [61]. Instead of computing reachable sets, we use zonotopes only to represent the set of means for reachable PD.

2) Efficient Solving Algorithm: Our idea is to (i) find the closest point $\boldsymbol{z}_k^* \in M_k$ to the strip center for each $k = 1, \dots, K$, (ii) construct a control sequence based on $\boldsymbol{z}_k^*$, and (iii) evaluate and choose the sequence which gives the highest probability of having the final state in the target strip. We explain the details as follows.

(i) Finding the Closest Zonotope Point to the Strip Center. Given a zonotope $Z = (c, \langle g_1, \dots, g_p\rangle)$ and a strip center hyperplane $H : \ell^T\boldsymbol{z} = \beta$, we propose Algorithm 2 to compute a group of scalars $\alpha_1, \dots, \alpha_p \in [-1, 1]$ to find a state $\boldsymbol{z}^* = \alpha_1 g_1 + \cdots + \alpha_p g_p \in Z$, which is closest to H among all $\boldsymbol{z} \in Z$. This algorithm can find a mean in $X_k(\mu_0)$, which provides the highest probability of reaching the target strip without solving an optimization problem. We can find the optimal mean z^* by pushing the center c towards the hyperplane as much as possible before reaching the hyperplane, i.e., let α to be 1 (Lines 12) if the generator towards the hyperplane (Line 9) otherwise -1 (Line 10). In the example of Case 2 of Fig. 5, all α are set to be 1 and -1, since they never reach the hyperplane. Once we find 1 or -1 is too much, i.e., z^* goes through the hyperplane, we can find a γ in the range of $(-1, 1)$ by computing $\gamma = (\beta - \ell^T\boldsymbol{z}^*)/(\ell^T g_i)$ (Line 4). In this case, z^* can fall onto the hyperplane, so we assign the value of γ to α_i (Line 5). Note that $\alpha_{i+1}, \cdots, \alpha_p$ are set to be 0 before termination since the algorithm takes the minimum control steps. An example is illustrated in Fig. 5 Case 1, where $\alpha_2 \in (-1, 1)$ and $\alpha_3 = 0$. This algorithm can be viewed as a modification of the zonotope/hyperplane intersection algorithm described in [62]; however, we aim to find the closest state in the zonotope to the hyperplane.

(ii) Constructing the Control Sequence. A control sequence can be constructed for reaching $\boldsymbol{z}_k^*$ during Algorithm 2. Given that Z is M_k, we may compute the control sequence $\boldsymbol{u}_0, \dots, \boldsymbol{u}_{k-1}$ using the returned scalars. To do so, assume that $\alpha_{i,j}$ is the computed scalar associated to the generator $A^i Bg_j^u$ for some $i = 0, \dots, k-1$ and $j = 1, \dots, m$. Then $\boldsymbol{u}_{k-i-1}$ can be obtained as $c_u + \sum_{j=1}^{m} \alpha_{i,1} g_j^u$. We may perform the above method for all $k = 1, \dots, K$, and choose the control sequence with the highest probability of reaching the target. We can now show that our solution is sound and complete.

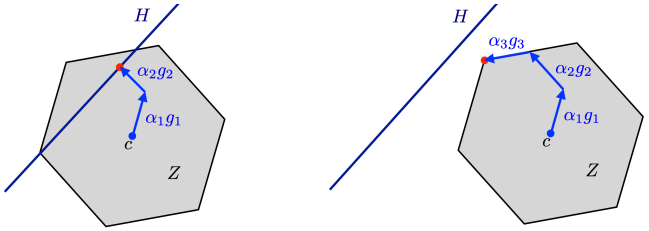


Fig. 5: Examples of the two cases of z^*. Case 1: $Z \cap H \neq \emptyset$, the optimal state z^* is found as $c + \alpha_1 g_1 + \alpha_2 g_2$. Case 2: $Z \cap H = \emptyset$, the optimal state $z^* = c + \alpha_1 g_1 + \alpha_2 g_2 + \alpha_3 g_3$ is a vertex of Z and is the closest state to H.

Algorithm 2 Finding a state $z^* \in Z$ such that $|\beta - \ell^T z^*| = \min\{|\beta - \ell^T z| \,|\, z \in Z\}$.

Input: $Z = (c, \langle g_1, \ldots, g_p \rangle)$, $H : \ell^T z = \beta$
Output: z^*, $\alpha_1, \ldots, \alpha_p$

1: $z^* \leftarrow c$; # starting from the center
2: $\alpha_1, \ldots, \alpha_p \leftarrow 0$; # initializing the parameters
3: **for** $i = 1$ to p **do**
4: **if** $\exists \gamma \in (-1, 1).(\ell^T(z^* + \gamma g_i) = \beta)$ **then**
5: $\alpha_i \leftarrow \gamma$; $\alpha_{i+1}, \cdots, \alpha_p \leftarrow 0$;
6: $z^* \leftarrow z^* + \alpha_i g_i$; # moving z^* onto H
7: break;
8: **else**
9: **if** $(\ell^T z^* > \beta \wedge \ell^T g_i > 0) \vee (\ell^T z^* < \beta \wedge \ell^T g_i < 0)$ **then**
10: $\alpha_i \leftarrow -1$; # moving z^* towards H by adding $-g_i$
11: **else**
12: $\alpha_i \leftarrow 1$; # moving z^* towards H by adding g_i
13: **end if**
14: $z^* \leftarrow z^* + \alpha_i g_i$;
15: **end if**
16: **end for**

Theorem IV.1. *Our approach is a sound and complete solution to the OPR problem.*

Soundness. We prove that the Gaussian distribution $\boldsymbol{x}_k \sim \mathcal{N}(z^*, \Sigma_k)$ produces the highest probability $\mathcal{P}(a \leq \ell^T \boldsymbol{x}_k \leq b)$ among all other distributions $\boldsymbol{x}_k \sim \mathcal{N}(z, \Sigma_k)$ with $z \in X_k(\mu_0)$. By Lemma IV.1, we only need to prove that z^* has the minimum distance to the center hyperplane $\ell^T \boldsymbol{x}_k = (a+b)/2$:

$$|(a+b)/2 - \ell^T z^*| = \min\{|(a+b)/2 - \ell^T z| \,|\, z \in X_k(\mu_0)\}.$$

To do so, we consider two cases. *Case 1:* $Z \cap H \neq \emptyset$. Then z^* must be in the hyperplane and we have that $(a+b)/2 - \ell^T z^* = 0$. Hence it is the minimum distance. *Case 2:* $Z \cap H = \emptyset$. We prove that z^* is the closest state in Z to the hyperplane. Algorithm 2 reduces the value $|\beta - \ell^T z^*|$ (Line 9-14) by the maximum extent in each iteration, then the value of $|\beta - \ell^T z^*|$ obtained from the resulting z^* must be the minimum, and therefore $|(a+b)/2 - \ell^T z^*|$ is the minimum. Fig. 5 gives an illustration of the two cases. Hence, if we compute all of the optimal PD for $k = 1, \ldots, K$, the best probability as well as the control sequence to the problem (2) can be obtained.

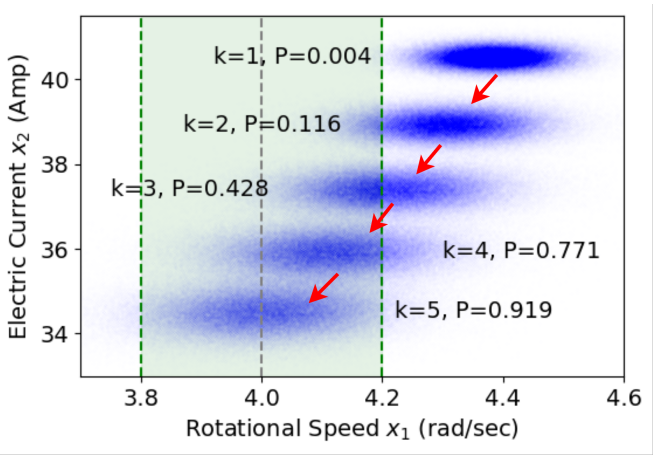


Fig. 6: The optimal reachable PD for DC motor speed benchmark. The green shaded area is the target set, where the center of the strip is marked in a grey dashed line. k: the number of recovery steps. P: the probability of reaching the target set. Red arrows indicate the time evolution. The proposed method gains the highest probability $P = 0.919$ at the 5^{th} step.

Completeness. We show that if there exists a solution to the problem (2), then it can always be found by our method. We consider the same two cases as those in the above proof.
Case 1: $Z \cap H \neq \emptyset$. The zonotope Z can be viewed as the result of bloating the center c by consecutively adding a centrally symmetric line segment determined by a generator. We define the intermediate set $S_j = \{c\} \oplus \bigoplus_{i=1}^{j} L_i$ wherein $L_i = \{\alpha_i g_i \,|\, \alpha_i \in [-1, 1]\}$, then $Z = S_p$. If $Z \cap H \neq \emptyset$, there must be some $0 \leq j \leq p-1$ such that $S_j \cap H = \emptyset$ but $S_{j+1} \cap H \neq \emptyset$, or $c \in H$. Both of these two cases can be found by Algorithm 2, since the algorithm always keeps z^* as one of the closest states to H. Thus, the first case can be detected by Line 4 in the $(j+1)$st iteration for some $\gamma \in [-1, 1]$, and the second case can be found in the first iteration as $\gamma = 0$.
Case 2: $Z \cap H = \emptyset$. When the intersection is empty, we have two cases for Z. (i) All of the states in Z satisfy the constraint $\ell^T z < \beta$, then Algorithm 2 (Line 9-14) computes a state z^* such that $\ell^T z^*$ equals to the support function $\rho_Z(\ell) = \sup\{\ell^T z \,|\, z \in Z\}$. (ii) All of the states in Z satisfy the constraint $\ell^T z > \beta$, it is analogous to the previous case, Algorithm 2 computes a state z^* such that $-\ell^T z^*$ equals to the support function $\rho_Z(-\ell) = \sup\{-\ell^T z \,|\, z \in Z\} = -\inf\{\ell^T z \,|\, z \in Z\}$. In both cases, z^* is the closest to H.
(iii) Computing the Maximum Probability. When an optimal mean z^* is computed for the kth step, the optimal PD is obtained as $\boldsymbol{x}_k \sim \mathcal{N}(z^*, \Sigma_k)$ wherein $\Sigma_k = A^k \Sigma_0 (A^k)^T + \Sigma_{\overline{w}_k}$. Then the probability of reaching the target set can be computed as $\mathcal{P}(a \leq \ell^T \boldsymbol{x}_k \leq b) = \frac{1}{2}\left(\text{erf}(\frac{b - \ell^T z^*}{\sqrt{2\ell^T \Sigma_k \ell}}) - \text{erf}(\frac{a - \ell^T z^*}{\sqrt{2\ell^T \Sigma_k \ell}})\right)$.

Example IV.1. *We consider the DC motor defined by*

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} -10 & 1 \\ -0.02 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \end{bmatrix} u$$

where x_1 denotes the rotational speed of the shaft, and

x_2 *denotes the electric current. The controller updates the voltage source* u *every 0.02 seconds to maintain the value of* x_1 *at* $4rad/s$*. The linear stochastic model is obtained by discretizing the above ODE using the control stepsize and adding two independent noises to* x_1, x_2*. Fig. 6 shows the optimal reachable PD computed by our method from the initial Gaussian distribution* $x_1 \sim \mathcal{N}(0,1)$ *and* $x_2 \sim \mathcal{N}(0,1)$*. The recovery target is defined by* $3.8 \leq x_1 \leq 4.2$ *rad/s.*

Computational Complexity of OPR. Given that the system (1) has n state variables, m control inputs, and d noises. Then A is an $n \times n$ matrix, B is an $n \times m$ matrix and C is an $n \times d$ matrix. We investigate the time complexity of computing the optimal recovery probability at the kth step. We assume that the operations in the basic arithmetic on reals have the time complexity $O(1)$ due to the use of floating-point numbers with fixed precision. We consider the multiplication of two matrices $M_1 \in \mathbb{R}^{m \times n}$ and $M_2 \in \mathbb{R}^{n \times k}$ costs $O(mnk)$. We provide Table II for the matrices that can be pre-computed and reused for each $k = 1, \ldots, K$.

We first evaluate the cost of computing the zonotope $X_k(\mu_0)$ wherein μ_0 is an n-dimensional vector representing the mean of the Gaussian distribution of $\boldsymbol{x}_0$. The set $\mathcal{U}$ of the control inputs is a box represented as a zonotope $(c_u, \langle g_1^u, \ldots, g_m^u \rangle)$ wherein $c_u, g_1^u, \ldots, g_m^u$ are all m-dimensional column vectors. Thus, the computation of $X_k(\mu_0)$'s center $A^k\mu_0 + \sum_{i=1}^{k} A^{i-1}Bc_u$ requires $O(n^2)$ to compute $A^k\mu_0$ and $O(kn)$ to compute the sum of vectors. Each of its km generators can be computed in $O(nm)$ and the total complexity is $O(knm^2)$. Hence, the computation of the $\mathcal{G}$-representation of $X_k(\mu_0)$ requires $O(n^2 + knm^2)$, and the zonotope has km generators.

Then we turn to the complexity of Algorithm 2. Given that Z is n-dimensional. The algorithm performs p iterations in each of which we verify the existence of $\gamma \in [-1, 1]$ that requires to compute two inner products $\ell^T \boldsymbol{z}^*$, $\ell^T g_i$ and evaluate the division $(\beta - \ell^T \boldsymbol{z}^*)/(\ell^T g_i)$. Thus, the time cost is $O(n)$. If such a γ does not exist, we use the two inner products computed previously to move $\boldsymbol{z}^*$ to a new position that is closer to the hyperplane, and it costs $O(n)$ (mainly cost by Line 14). Hence, the time complexity of Algorithm 2 with $Z = X_k(\mu_0)$ is $O(kmn)$ which is linear in the size of $X_k(\mu_0)$, since it has km generators which are all n-dimensional.

TABLE II: Pre-computed and reused matrices

Matrix	A^k	A^kB	A^kBc_u	$A^kBg_i^u$	A^kC	$\Sigma_{\overline{w}_k}$
Total Complexity of $k = 1, \ldots, K$	$O(Kn^3)$	$O(Kn^2m)$	$O(Knm^2)$	$O(Knm^2)$	$O(Kn^2d)$	$O(Kn^2d)$

Finally, we consider the evaluation of the best probability of reaching the target w.r.t. the distribution $\boldsymbol{x}_k \sim \mathcal{N}(\boldsymbol{z}^*, \Sigma_k)$. The computation of Σ_k requires $O(n^3)$ when $\Sigma_{\overline{w}_k}$ is pre-computed, and linear mapping the Gaussian distribution to $(\mathcal{N}(\ell^T \boldsymbol{z}^*, \ell^T \Sigma_k \ell))$ costs $O(n^2)$. Assuming that the time cost of evaluating the error function $\mathrm{erf}(\cdot)$ is $\mathcal{C}_{\mathrm{erf}}$, the time cost of the probability evaluation is $O(n^3 + \mathcal{C}_{\mathrm{erf}})$. The total complexity of finding a solution for the optimization problem (2) is

$$O(K(n^3 + n^2m + nm^2 + n^2d) + K^2nm^2 + K\mathcal{C}_{\mathrm{erf}}) \quad (6)$$

which is at most *cubic* in the number of state variables, *quadratic* in the number of control inputs and K, and linear in the complexity of computing error functions. When Algorithm 2 is used online, all of the matrices in Table II can be pre-computed and kept in hash tables. Therefore, the online computational complexity becomes $O(K(n^2 + Knm^2 + n^3 + \mathcal{C}_{\mathrm{erf}}))$.

D. Algorithms for Other Components

1) Supporting Components: **Attack detector.** The existing attack detectors may have two levels of capabilities. (D1) Detect whether there is an anomaly (e.g., the physical behavior of the system is not consistent with the intended control actions). Since we cannot trust any sensor data after detection, the recovery cannot get any sensor feedback, also known as *Open-Loop (OL)* Recovery. D1 captures the case when all sensors are compromised. (D2) Detect an anomaly and identify which sensors are receiving the malicious data (e.g., detect that the IMU is sending erroneous data, while the LiDAR is reliable) [63], [64]. The recovery can leverage good sensor data as feedback, also known as *Partial Closed-Loop (PCL)* Recovery. We will implement and evaluate Open-Loop Recovery and Partial Closed-Loop Recovery in our evaluation section and discuss the pros and cons of each strategy there.
Checkpointer. We also consider that a Checkpointer already exists since it is the focus of some existing work, such as [13]. It can record historical state estimates $\boldsymbol{x}$, sensor data $\boldsymbol{y}$, and control input $\boldsymbol{u}$ within a sliding window. The window is typically larger than the maximum detection delay, providing a trustworthy state $\boldsymbol{x}_w$ insusceptible to sensor attacks.

2) State Reconstruction: The State Reconstruction component estimates the PD of a system state using historical data. There are two phases: predict and update. It can work with either of these two detector options above. The predict phase computes the predicted current state estimate and covariance matrix from the previous ones. If there is a detector of type (D1), we use the mathematical model to estimate the system state and its covariance matrix. If there is a detector of type (D2), it can identify good sensor data. Thus, the update phase improves the estimate based on trustworthy sensor data. We use an extended Kalman filter (EKF) and the good sensor data $\boldsymbol{y}'_{w+1}, \ldots, \boldsymbol{y}'_r$ to obtain the probability distribution. We modify line 3 from Algorithm 1 to $\mathcal{N}_i \leftarrow \mathrm{StateReconstruction}(\boldsymbol{x}_w, \boldsymbol{u}_w, \ldots, \boldsymbol{u}_{r-1}, \boldsymbol{y}'_{w+1}, \ldots, \boldsymbol{y}'_r)$, which runs the EKF. The experiments show that our open-loop recovery can safely recover various CPS with a detector of type (D1), and that the closed-loop recovery significantly enhances performance if we have a detector of type (D2).

3) Model Adaptation: The Model Adaptation online locally linearizes and discretizes the nonlinear model into a discrete-time linear model, which is a common routine in the control community. We use Taylor Series expansion to linearize nonlinear systems around the current state estimate and control input and then use the Euler method to discretize it. In this

way, the Model Adaptation obtains a discrete-time LTI model in the form of Equation (3), and the OPR algorithm can work on it. Since we perform linearization and discretization in every control step, the approximation error is rather small. Experiments show that this approximation does not negatively impact the efficacy of the OPR method.

V. Evaluation

We now compare the performance of our proposed recovery strategy with previous work. We test several use cases, including a drone, a robotic vehicle, a simulated vehicle, and several linear systems. We will show that the Open-Loop recovery outperforms the previous works and then present the Partial-Closed Loop recovery as a further improvement. The code is available at this link and integrated into the CPSim [65].

A. Experimental Settings

This subsection explains the example systems, metrics, and baseline recovery strategies.

1) Implementation details: We implement the recovery strategies in systems with linear and nonlinear dynamics. We first consider a drone simulated in high-fidelity simulators, Gazebo and Airsim. We also use two ground vehicles, one simulated in a high-fidelity simulator, SVL, and a real-world robot. Finally, we simulate several linear systems. Next, we present the implementation details of each use case and define the target sets, which are application-specific [51].

Drone from Gazebo/AirSim simulator.

We simulate the drone using Gazebo and Airsim as the physics providers [66]. We use the group of all rotation matrices in $\mathbb{R}^3$, denoted as $SO(3)$, to model the drone behavior. The following model describes the drone dynamics [67], [68],

$$\begin{aligned} \dot{x} &= v & \dot{R} &= R\hat{\Omega} \\ m\dot{v} &= mge_3 - fRe_3 & J\dot{\Omega} &= U - \Omega \times J\Omega, \end{aligned} \tag{7}$$

where m is the drone mass, g the gravity, $J \in \mathbb{R}^{3\times3}$ is the inertia matrix, $x \in \mathbb{R}^3$ is the drone position, $v \in \mathbb{R}^3$ is the drone velocity, $\Omega \in \mathbb{R}^3$ is the drone angular velocity, $R \in SO(3)$ is the attitude, and e_3 is a vector that defines the inertial frame. The operator $\times$ is the cross product, and $\hat{}$ is an operator such that $\hat{x}y = x \times y$ and $\hat{x}^T = -\hat{x}$, for $x, y \in \mathbb{R}^3$ (see [68], [69] for details). The inputs $f \in \mathbb{R}$ and $U \in \mathbb{R}^3$ are the total thrust and the resultant moment generated by four rotors. We assume that each motor generates a torque proportional to the thrust, that the distance between the quadrotor gravity center and each rotor is d, and that the mass distribution of the drone is symmetric. Thus, the relationship between the model inputs and the motors' thrust is linear (see [68], [69]), and the inertia matrix J is diagonal and positive definite.

The drone uses the controller in [67] when there is no alarm. We design the strip parameters as follows to avoid an unauthorized landing. The parameter ℓ is one in its third component, corresponding to the drone height, and zero in the others. Before the attack begins, the drone hovers at a height of $10\,m$, so the remaining strip values are $a = 9.8, b = 10.2$. The attacker compromises the drone's height sensor to produce an unauthorized landing, which is detected $1.5\,s$ later.

Vehicle from SVL Simulator. Autonomous vehicles perform lateral control to track the path the planning module provides. The vehicle can be approximately modeled with a nonlinear kinematic model [20] given by,

$$\dot{x} = v\cos(\theta + \beta), \quad \dot{y} = v\sin(\theta + \beta), \quad \dot{\theta} = L^{-1}v\tan(\delta)\cos(\beta),$$

where x, y are the position in a two-dimensional space, θ is the angle with respect to x axis, δ is the steering angle, v is the vehicle velocity, $\beta = \arctan\left(l_r \frac{\tan(\delta)}{L}\right)$ with l_r the distance between the rear axle and the vehicle center of mass, and L is the length between the front and rear wheels.

We use a vehicle in SVL, a unity-based high-fidelity simulator for autonomous driving (AD). The nominal controller is the Stanley lateral controller in [70], which requires measurements from an IMU and GPS. The vehicle's mechanical structure constraints the control input to $[-30^\circ, 30^\circ]$. We want the vehicle to drive to the shoulder of the road and stop there when recovery ends. Thus, the target set is the road shoulder, expressed as a strip with parameters $l = [0, 1, 0]$, $a = 45.95$. We simulate an IMU sensor attack in which the attacker can alter the sensor values through acoustic signals [71]. The attack drives the vehicle to enter the oncoming traffic lane, and the detector can identify it after $1.5\,s$.

Robotic vehicle Testbed (RV). We implement a similar setup to the vehicle from SVL in a real RV. The RV is based on the BARC project [72]. It uses ROS 1 Noetic with Python 3 and has two motors to control the vehicle's velocity and direction. This vehicle has the same model as the vehicle from SVL.

For this experiment, the vehicle has to follow a straight line at a constant velocity v. While the detector does not trigger an alarm, the vehicle uses a proportional–integral–derivative (PID) controller to obtain the steering angle δ. The RV's physical characteristics constrain the steering angle to be in the range of $[-60^\circ, 60^\circ]$. As the vehicle from SVL, we want the RV to rest on the shoulder at the end of the recovery. Then, the strip parameters are $\ell = [0, 1, 0]$, $a = -0.3, b = -0.7$, which corresponds to the road shoulder. The attacker compromises the sensor that measures the RV's position to steer it to the opposite lane, and the detector raises the alarm $1.5\,s$ later.

Linear Systems. Finally, we use systems with linear dynamics and linearized versions of systems to show that our strategy also outperforms the baselines for simpler systems. Table III summarizes the systems and the number of states and inputs. We omit the particular parameters for space constraints.

TABLE III: Linear benchmarks

Benchmark	Number of inputs	Number of states
DC motor speed [73]	1	2
Aircraft Pitch [74]	1	3
Boeing 747 [75]	1	5
Quadrotor [76]	1	12
F16 [77]	2	4
Quadruple tank [78]	2	4

2) Metrics for evaluation: We now present the performance metrics to compare the strategies. We run several experiments to obtain the metrics as we consider stochastic systems.

Successful Recovery Rate (SRR). An attack recovery is successful if the system state is in the target set at the end of recovery. We compute SRR as the ratio of successful recoveries to the total number of experiments.

State distance to the center of the target set. We want the system to be as close to the target set at the end of the reconfiguration. Therefore, we compute the distance from the system's state at the end of the recovery to the target set center.

Recovery time. We want the system under attack to go to the target set as fast as possible. We measure the number of control steps that a recovery strategy requires to steer the system to the target set. The metric evaluates the recovery speed.

Computational overhead. Since the computational resource is typically limited in CPSs, an online recovery strategy should have low computational overhead. We measure the time a strategy requires to find the recovery actions in each iteration. This metric evaluates the strategies' complexity.

3) Baselines: In Section II, we offer a comprehensive comparison of related work. This section compares our proposed open-loop OPR strategy (detailed in Section IV-D2) with strategies employing backup controllers and virtual sensors. For backup controllers, we implement an LQR-based real-time recovery, as presented in [18], an advancement over [15]. This strategy considers safety guarantees and real-time requirements, a feature absent in [35]. Regarding virtual sensors, we use a strategy informed by previous studies [12], [14], [38].

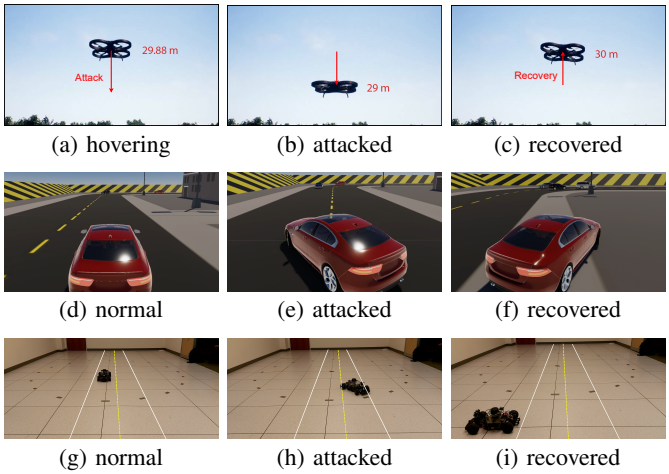
LQR-based real-time recovery (RTR-LQR). After the attack detection, the method in [18] formulates the recovery as a linear-quadratic programming problem whose solution provides a sequence of control actions to recover the system. To make the method applicable to nonlinear systems, we linearize the nonlinear dynamics around the reconstructed state and control input when recovery begins.

Virtual sensor recovery (VS). The strategy replaces corrupted sensor data with model-predicted data from a virtual sensor. The nominal controller leverages virtual sensor data to correct systems' behavior during an attack. To make a fair comparison, we improve such strategies by adopting a checkpointing protocol to provide trustworthy states. We measure all metrics when VS reports that the system is inside the target set.

4) Experimental Environment: Simulations run on a computer powered by an NVIDIA T600 GPU, an 11th Generation Intel Core I7-11800H, and 32GB RAM memory. The real-world implementation on an RV runs on an Odroid XU4 board.

B. Experimental results

We summarize our results of OPR-OL in Figs. 7 and 8, which exemplify an execution of the recovery strategy, and Tables IV and V. The first table presents the performance of each baseline in terms of the SRR. The second table summarizes the recovery time, computation time, and distance to the strip. As we deal with stochastic systems, we run several simulations and implementation executions and summarize the aggregate statistics in the tables and plots. In the remainder of this section, we discuss the results of implementing the strategies on each benchmark.



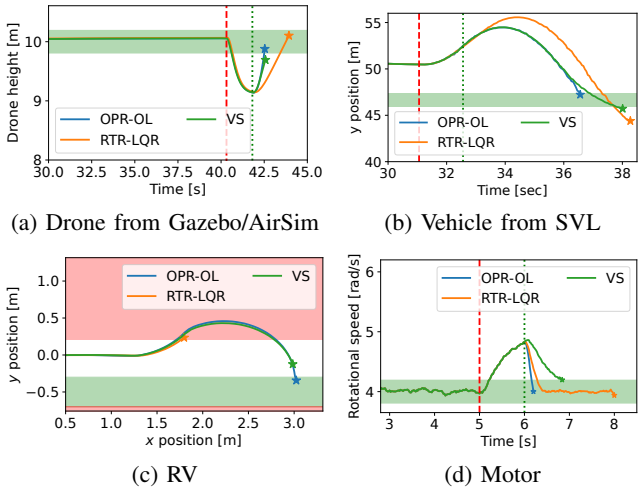
(a) hovering (b) attacked (c) recovered

(d) normal (e) attacked (f) recovered

(g) normal (h) attacked (i) recovered

Fig. 7: Recovery examples: drone, SVL, and RV.

1) Recovery Effect: We first intuitively demonstrate the execution of different attack-recovery proposals using time-series state plots, and then we compare these strategies using the SRR and distances to the target set center.



(a) Drone from Gazebo/AirSim (b) Vehicle from SVL

(c) RV (d) Motor

Fig. 8: Comparison of the system executions using different recovery strategies. Stars mark the final system states of recovery. The green shaded area is the target set.

Demonstration of Recovery Processes. Fig. 7 demonstrates the recovery process of OPR-OL. First, Figs. 7a, 7d and 7g show the vehicles are operating without attack: the drone is at a safe height, and the ground vehicles are following the lane path. The attacker then compromises the vehicles' sensors to provoke an unauthorized drone landing and make the SVL vehicle and the RV enter the opposite lane as Figs. 7b, 7e, 7h show. Finally, the attack recovery drives the drone back to a safe height and parks the vehicles on the safe road shoulder to avoid dangerous consequences such as crashes with incoming traffic (see Figs. 7c, 7f and 7i).

Fig. 8 shows the trajectory of the recovery process and illustrates how OPR-OL compares to previous work (RTR-LQR and VS). For this Figure, we include one linear benchmark (Fig. 8(d)). The blue, orange, and green curves represent OPR-OL, RTR-LQR, and VS strategies, respectively. The dashed red vertical line indicates the onset of sensor attacks, and the dotted green vertical line indicates their detection. In addition, the green shaded area marks the target set, the target strip. When the attack is detected, we trigger the recovery methods to take control of the system.

Fig. 8 also shows that our proposal (i) steers the system states into the target set (ii) and recovers system states faster than previous work for linear and nonlinear systems. In contrast, the baselines cannot steer all the benchmarks to the target set. Particularly, VS leaves the system outside the target set, even for the linear system, while the RTR-LQR cannot recover the ground vehicles (see Figs. 8b and 8d). Moreover, the RV exposes the limitations of previous works, as the RTR-LQR cannot find a solution to the recovery problem and leaves the vehicle in the opposite lane (see Fig. 8c).

Successful recovery rate. We present the SRR in Table IV. Our strategy always achieves an SRR higher than the baselines. The OPR-OL achieved an SRR of 80% or more. In contrast, the VS only achieved a maximum SRR of 20% in the nonlinear systems and 55% in the linear systems. The RTR-LQR achieved an SRR higher than 50% for the linear systems and the drone, but this strategy could not steer the vehicles (in SVL and the RV) to the target set. The optimization problem that it solves becomes unfeasible for the RV. Consequently, RTR-LQR does not find a sequence of inputs to recover the system, leaving the system in control of the attacker.

TABLE IV: Successful Recovery rate for each benchmark

Benchmark	VS	RTR-LQR	OPR-OL
Drone	0.00	**80.00**	**80.00**
SVL	20.00	0.00	**100.00**
Robotic vehicle	0.00	0.00	**80.00**
Linear systems	55.12	53.96	**88.12**

State distance to the center of the target set. As shown in Table IV, OPR-OL gets the system closer to the center of the target set than the baselines for all the benchmarks. RTR-LQR presents a similar performance to OPR-OL in the drone system. However, RTR-LQR presents the worst performance for ground vehicles, leaving them far from the target set. VS performs better on the vehicles but does not improve OPR-OL. Although the VS strategies can decrease the effectiveness of the attacks, they cannot accurately provide the instant when the system is safe and arrives at the target set. Thus, the system might not be inside the target set when the VS indicates it.

2) Recovery Speed: Table V shows the time each strategy requires to steer the system to the target set. RTR-LQR is the strategy that requires more time. In contrast, our strategy is the fastest for the drone, the vehicle from SVL, and the linear systems. VS requires fewer time steps for the RV than the other strategies. However, this is, in fact, a disadvantage of VS. For those systems, VS indicates that the system is in the target set when the vehicle has not entered yet and makes an early stop. Thus, the RV stops in the lane center, which might have more traffic.

3) Computational Overhead: Based on Table V, we confirm that VS is a fast strategy that imposes a small computational overhead but cannot recover the system as discussed above. Additionally, the computational overhead of RTR-LQR is similar for systems with few states and inputs as the SVL vehicle but increases with the number of states and inputs.

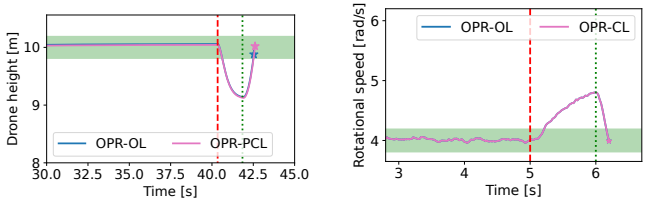
Our strategy requires an acceptable computational overhead and is implementable in real systems with limited resources while improving the SRR. On average, OPR-OL requires $8.33\,ms$ to solve the recovery problem. Particularly, the real RV requires on average less than $9\,ms$ to compute the recovery, while the vehicle needs a control action every $100\,ms$.

4) False alarms: An incorrect call to an attack-recovery procedure (e.g., due to a false alarm by the intrusion detection system) should not make the system unsafe. We obtain the SRR for the drone and the linear systems with all the strategies after a false alarm. We find that the drone always reaches the desired height after a false alarm using the OPR-OL. For the linear systems cases, OPR-OL can steer them to the target set in 97% of the cases, while RTR-LQR and VS obtain an SRR of 90%. Then, OPR-OL preserves the system safe and outperforms the baselines, even after an incorrect call.

C. Partial Closed-Loop Recovery

We have shown that OPR-OL can recover the systems and outperform the baseline strategies. Next, we will show that we can further improve our strategy by incorporating sensors that are not under attack.

Fig. 9 presents one run of the attack recovery for the drone and the linear systems using the OPR-OL (blue line) and the OPR-PCL (purple line); the results for the other systems are similar but omitted for space constraints. The drone arrives closer to the desired height with the OPR-PCL, while both strategies perform similarly for the motor. Additionally, we obtain the SRR for all the benchmarks, which increases for all the systems: the OPR-PCL successfully steers the system RV, drone, and SVL vehicle to the target set in $100\,\%$ of the simulations, and the SRR increases to $96.04\,\%$ for the linear systems. This indicates that the OPR-PCL can incorporate data from sensors that are not under attack and enhance the OPR.



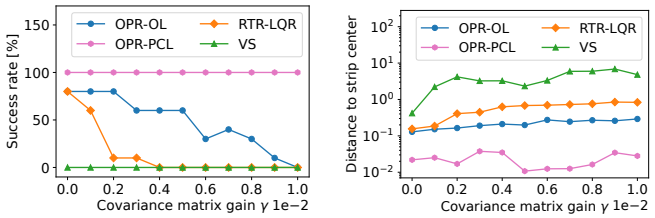
(a) Drone from Gazebo/AirSim (b) Motor

Fig. 9: Comparison of the OPR-OL and OPR-CL approaches.

TABLE V: Performance metric throughout several experiments. Values are the average ± one standard deviation

Benchmark	Distance to strip center			Recovery time [steps]			Computation time [ms]		
	VS	RTR-LQR	OPR-OL	VS	RTR-LQR	OPR-OL	VS	RTR-LQR	OPR-OL
Drone	0.42 ± 0.11	0.16 ± 0.13	$\mathbf{0.13 \pm 0.04}$	36.30 ± 4.12	90.40 ± 16.31	$\mathbf{34.40 \pm 0.92}$	$\mathbf{0.63 \pm 0.12}$	1.88 ± 16.75	1.49 ± 3.30
SVL	0.89 ± 0.10	2.79 ± 0.54	$\mathbf{0.54 \pm 0.08}$	109.50 ± 0.81	101.00 ± 0.00	$\mathbf{81.50 \pm 0.50}$	$\mathbf{0.74 \pm 0.30}$	4.52 ± 54.28	8.83 ± 3.36
Robotic vehicle	0.37 ± 0.02	0.73 ± 0.01	$\mathbf{0.16 \pm 0.04}$	$\mathbf{26.00 \pm 0.00}$	–	30.50 ± 0.67	$\mathbf{0.45 \pm 0.05}$	–	8.33 ± 3.73
Linear systems	0.25 ± 0.24	0.31 ± 0.33	$\mathbf{0.11 \pm 0.14}$	33.01 ± 21.14	39.75 ± 37.07	$\mathbf{11.50 \pm 3.14}$	$\mathbf{0.07 \pm 0.04}$	11.15 ± 12.79	0.61 ± 0.20

1) Impact of Uncertainty: We study the effect of the stochastic process on recovery strategies. We use the drone setup for this study with the same characteristics we presented before, but we add noise that may model unknown elements such as wind. The noise covariance matrix is given by $C = \gamma I$; a larger γ means there is more uncertainty. Fig. 10a presents the SRR of maintaining the desired height and entering the target set, and Fig. 10b the distance to the target set center. The SRR decreases for all the open-loop strategies as there is more uncertainty, but the SRR of the OPR-OL is always higher than the baseline strategies' SRR. Additionally, the OPR-OL always steers the drone closer to the desired height than the baselines. Finally, we corroborate that we can further improve our strategy by using uncompromised sensors. The OPR-PCL achieves a $100\,\%$ success for all noise levels, and the distance to the target set center is always closer to zero. Thus, we conclude our strategies can handle noise better than the baselines, even for nonlinear systems.



(a) Success rate (b) Distance to strip center

Fig. 10: Success rate and average distance to the target set center with increasing noise for the drone.

D. Discussion

Partial closed-loop vs. open-loop OPR. OPR-PCL can incorporate the measurements of trustworthy sensors into the recovery at each step. Therefore, OPR-PCL can reject the uncertainty that noise introduces and find a recovery that steers the system closer to the center of the target set. Although OPR-PCL is an improvement over OPR-OL for systems with linear and nonlinear dynamics, we find that nonlinear systems can benefit more from OPR-PCL than linear systems. Nonlinear systems have more complex dynamics that are more difficult to capture with mathematical models, and introducing trustworthy measurements alleviates the model mismatches.

Completeness. As we show theoretically and through simulations, the proposed approach always produces a recovery control sequence to maximize the probability of arriving at the target set. In contrast, RTR-LQR could not find a solution in several situations. Consequently, the system cannot react to the attack, leaving the control to the attacker.

Computation time trade-off. For a given system, the parameter K of our strategy imposes a trade-off between the computation time and the probability of arriving at the target set. A small K introduces a small computational overhead, but it might not give enough time for the system to arrive at the target set. However, a large K will increase the computation time, while the probability of arriving at the target set will not necessarily increase due to noise accumulation.

Limitations. Our strategy requires an attack detector in place to trigger it. Moreover, OPR-PCL requires a detector that identifies good sensors, which is an additional assumption. Although improving the performance of the detectors is out of scope, detectors may affect the performance of attack recovery. For instance, if the detector cannot identify the attack, the recovery is not activated, leaving control to the attacker. However, some research is focused on dealing with such stealthy attacks [79]. Additionally, our strategy provides an efficient solution to the recovery problem when the target set is a strip. If the target set cannot be expressed as a strip, it requires an optimization solver to find a recovery control sequence, which potentially makes the solution time-consuming. Besides, discretization and linearization in model adaptation introduce uncertainties, which can be considered in the system model.

Finally, as we focused on attacks against sensors, our strategy requires that the device that computes the recovery is not under attack. If the attacker manages to compromise the controller and modify the computations, our strategy cannot respond. However, that is out of the scope of this paper.

VI. Conclusions

In this paper, we introduce a new algorithm to recover CPS from sensor attacks. First, we identify limitations of state-of-the-art recovery mechanisms, then we propose a new formal framework to formulate the problem and prove that under certain conditions, our proposed algorithm is sound (all solutions maximize the probability of recovery), complete (we always find a solution, unlike previous work), and efficient. We then show, through multiple high-fidelity simulations and with real-world robotic vehicles, that our solution is effective and efficient for nonlinear systems. Our experimental simulations in various scenarios show how our proposed recovery algorithms outperform previous work. As attack-detection systems improve over time, the next natural step is to include autonomous attack responses. We hope that this paper will motivate more work in this area.

ACKNOWLEDGEMENTS

This work was supported in part by NSF CNS-1929410, CNS-1931573, CNS-2333980, the Air Force under PIA FA8750-19-3-1000, the CAHSI-Google Institutional Research Program, and by the National Center for Transportation Cybersecurity and Resiliency (TraCR) (a U.S. Department of Transportation National University Transportation Center).

The U.S. Government is authorized to reproduce and distribute copies for Governmental purposes notwithstanding any copyright or other restrictive legends. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force or the U.S. Government the Department of Transportation, and National Science Foundation (NSF).

REFERENCES

[1] C. Yan, H. Shin, C. Bolton, W. Xu, Y. Kim, and K. Fu, "Sok: A minimalist approach to formalizing analog sensor security," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 480–495.

[2] G. Y. Dayanıklı, S. Sinha, D. Muniraj, R. M. Gerdes, M. Farhood, and M. Mina, "{Physical-Layer} attacks against pulse width {Modulation-Controlled} actuators," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 953–970.

[3] H. Sathaye, M. Strohmeier, V. Lenders, and A. Ranganathan, "An experimental study of {GPS} spoofing and takeover attacks on {UAVs}," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 3503–3520.

[4] D. U. Case, "Analysis of the cyber attack on the ukrainian power grid," *Electricity Information Sharing and Analysis Center (E-ISAC)*, vol. 388, pp. 1–29, 2016.

[5] Chrysler recalls 1.4m vehicles for bug fix: https://www.wired.com/2015/07/jeep-hack-chrysler-recalls-1-4m-vehicles-bug-fix/. [Online]. Available: https://www.wired.com/2015/07/jeep-hack-chrysler-recalls-1-4m-vehicles-bug-fix/

[6] A. H. Rutkin, "spoofers use fake gps signals to knock a yacht off course," *MIT Technology Review*, 2013.

[7] Slammer worm crashed ohio nuke plant network: https://www.securityfocus.com/news/6767. [Online]. Available: https://www.securityfocus.com/news/6767

[8] J. P. Farwell and R. Rohozinski, "Stuxnet and the future of cyber war," *Survival*, vol. 53, no. 1, pp. 23–40, 2011.

[9] M. A. Arroyo, M. T. I. Ziad, H. Kobayashi, J. Yang, and S. Sethumadhavan, "Yolo: frequently resetting cyber-physical systems for security," in *Autonomous Systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2019*, vol. 11009. International Society for Optics and Photonics, 2019, p. 110090P.

[10] F. Abdi, C.-Y. Chen, M. Hasan, S. Liu, S. Mohan, and M. Caccamo, "Guaranteed physical security with restart-based design for cyber-physical systems," in *Proceedings of the ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018.

[11] L. Niu, D. Sahabandu, A. Clark, and P. Radha, "Verifying safety for resilient cyber-physical systems via reactive software restart," in *accepted) 2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*, 2022.

[12] A. A. Cardenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: risk assessment, detection, and response," in *Proceedings of the 6th ACM symposium on information, computer and communications security*, 2011, pp. 355–366.

[13] F. Kong, M. Xu, J. Weimer, O. Sokolsky, and I. Lee, "Cyber-physical system checkpointing and recovery," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 22–31.

[14] H. Choi, S. Kate, Y. Aafer, X. Zhang, and D. Xu, "Software-based realtime recovery from sensor attacks on robotic vehicles," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 349–364.

[15] L. Zhang, X. Chen, F. Kong, and A. A. Cardenas, "Real-time recovery for cyber-physical systems using linear approximations," in *41st IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2020.

[16] S. Mohan, S. Bak, E. Betti, H. Yun, L. Sha, and M. Caccamo, "S3a: secure system simplex architecture for enhanced security of cyber-physical systems," *arXiv preprint arXiv:1202.5722*, 2012.

[17] M. Liu, L. Zhang, V. V. Phoha, and F. Kong, "Learn-to-respond: Sequence-predictive recovery from sensor attacks in cyber-physical systems," in *2023 IEEE Real-Time Systems Symposium (RTSS)*, ser. RTSS '23, 2023, pp. 78–91.

[18] L. Zhang, P. Lu, F. Kong, X. Chen, O. Sokolsky, and I. Lee, "Real-time attack-recovery for cyber-physical systems using linear-quadratic regulator," in *21st ACM SIGBED International Conference on Embedded Software (EMSOFT)*, 2021.

[19] L. Zhang, K. Sridhar, M. Liu, P. Lu, X. Chen, F. Kong, O. Sokolsky, and I. Lee, "Real-time data-predictive attack-recovery for complex cyber-physical systems," in *2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2023, pp. 209–222.

[20] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin, "SAVIOR: Securing autonomous vehicles with robust physical invariants," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

[21] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg, "Limiting the impact of stealthy attacks on industrial control systems," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1092–1105.

[22] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell, "A survey of physics-based attack detection in cyber-physical systems," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[23] F. Akowuah and F. Kong, "Physical invariant based attack detection for autonomous vehicles: Survey, vision, and challenges," in *The Fourth International Conference on Connected and Autonomous Driving (MetroCAD 2021)*. IEEE, 2021.

[24] Y. Luo, Y. Xiao, L. Cheng, G. Peng, and D. Yao, "Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–36, 2021.

[25] B. Nassi, R. Bitton, R. Masuoka, A. Shabtai, and Y. Elovici, "Sok: Security and privacy in the age of commercial drones," in *42nd IEEE Symposium on Security and Privacy, SP 2021*. Institute of Electrical and Electronics Engineers Inc., 2021, pp. 1434–1451.

[26] M. Foley, C. Hicks, K. Highnam, and V. Mavroudis, "Autonomous network defence using reinforcement learning," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 1252–1254.

[27] A. Molina-Markham, C. Miniter, B. Powell, and A. Ridley, "Network environment design for autonomous cyberdefense," *arXiv preprint arXiv:2103.07583*, 2021.

[28] L. Burbano, K. Garg, S. J. Leudo, A. A. Cardenas, and R. G. Sanfelice, "Online attack recovery in cyberphysical systems," *IEEE Security & Privacy*, 2023.

[29] F. Akowuah and F. Kong, "Recovery-by-learning: Restoring autonomous cyber-physical systems from sensor attacks," in *27th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 2021.

[30] K. Vatanparvar and M. A. Al Faruque, "Self-secured control with anomaly detection and recovery in automotive cyber-physical systems," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 788–793.

[31] T. L. Crenshaw, E. Gunter, C. L. Robinson, L. Sha, and P. Kumar, "The simplex reference model: Limiting fault-propagation due to unreliable components in cyber-physical system architectures," in *28th IEEE International Real-Time Systems Symposium (RTSS)*. IEEE, 2007, pp. 400–412.

[32] X. Wang, N. Hovakimyan, and L. Sha, "L1simplex: fault-tolerant control of cyber-physical systems," in *2013 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2013, pp. 41–50.

[33] K. Zhou and J. C. Doyle, *Essentials of robust control*. Prentice hall Upper Saddle River, NJ, 1998, vol. 104.

[34] M. Green and D. J. Limebeer, *Linear robust control*. Courier Corporation, 2012.

[35] P. Dash, G. Li, Z. Chen, M. Karimibiuki, and K. Pattabiraman, "Pidpiper: Recovering robotic vehicles from physical attacks," in *2021 51st*

Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2021, pp. 26–38.

[36] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annual Reviews in Control*, vol. 46, pp. 8–28, 2018.

[37] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake, "Lyapunov-stable neural-network control," *arXiv preprint arXiv:2109.14152*, 2021.

[38] K. Paridari, N. O'Mahony, A. E.-D. Mady, R. Chabukswar, M. Boubekeur, and H. Sandberg, "A framework for attack-resilient industrial control systems: Attack detection and controller reconfiguration," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 113–128, 2017.

[39] P. Zuliani, "Statistical model checking for biological applications," *Int. J. Softw. Tools Technol. Transf.*, vol. 17, no. 4, pp. 527–536, 2015.

[40] A. P. Vinod, J. D. Gleason, and M. M. K. Oishi, "Sreachtools: a MATLAB stochastic reachability toolbox," in *Proc. of HSCC'19*. ACM, 2019, pp. 33–38.

[41] Y. Wang, M. Zarei, B. Bonakdarpour, and M. Pajic, "Statistical verification of hyperproperties for cyber-physical systems," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 5s, pp. 1–23, 2019.

[42] S. Sun, Y. Zhang, X. Luo, P. Vlantis, M. Pajic, and M. M. Zavlanos, "Formal verification of stochastic systems with relu neural network controllers," in *Proc. of ICRA'22*. IEEE, 2022, pp. 6800–6806.

[43] A. J. Thorpe, V. Sivaramakrishnan, and M. M. K. Oishi, "Approximate stochastic reachability for high dimensional systems," in *2021 American Control Conference (ACC)*, 2021, pp. 1287–1293.

[44] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems," *Automatica*, vol. 44, no. 11, pp. 2724–2734, 2008.

[45] S. Summers and J. Lygeros, "Verification of discrete time stochastic hybrid systems: A stochastic reach-avoid decision problem," *Automatica*, vol. 46, no. 12, pp. 1951–1961, 2010.

[46] A. P. Vinod and M. M. Oishi, "Stochastic reachability of a target tube: Theory and computation," *Automatica*, vol. 125, p. 109458, 2021.

[47] T. He, L. Zhang, F. Kong, and A. Salekin, "Exploring inherent sensor redundancy for automotive anomaly detection," in *57th Design Automation Conference*. ACM, 2020.

[48] L. Zhang, Z. Wang, M. Liu, and F. Kong, "Adaptive window-based sensor attack detection for cyber-physical systems," in *59th Design Automation Conference*. ACM, 2022.

[49] Z. Wang, L. Zhang, Q. Qiu, and F. Kong, "Catch you if pay attention: Temporal sensor attack diagnosis using attention mechanisms for cyber-physical systems," in *2023 IEEE Real-Time Systems Symposium (RTSS)*, ser. RTSS '23, 2023, pp. 64–77.

[50] L. Zhang, M. Liu, and F. Kong, *AI-enabled Real-Time Sensor Attack Detection for Cyber-Physical Systems*, ser. Book. Cham: Springer International Publishing, 2023, pp. 91–120. [Online]. Available: https://doi.org/10.1007/978-3-031-42637-7_6

[51] K. Hobbs, M. Mote, M. Abate, S. Coogan, and E. Feron, "Run time assurance for safety-critical systems: An introduction to safety filtering approaches for complex control systems," *arXiv preprint arXiv:2110.03506*, 2021.

[52] A. Lavaei, M. Khaled, S. Soudjani, and M. Zamani, "AMYTISS: parallelized automated controller synthesis for large-scale stochastic systems," in *Proc. of CAV'20*, ser. LNCS, vol. 12225. Springer, 2020, pp. 461–474.

[53] A. J. Thorpe and M. Oishi, "SOCKS: A stochastic optimal control and reachability toolbox using kernel methods," in *Proc. of HSCC'22*. ACM, 2022, pp. 21:1–21:12.

[54] S. E. Z. Soudjani, C. Gevaerts, and A. Abate, "FAUST ^{2} : Formal abstractions of uncountable-state stochastic processes," in *Proc. of TACAS'15*, ser. LNCS, vol. 9035. Springer, 2015, pp. 272–286.

[55] F. Shmarov and P. Zuliani, "Probreach: A tool for guaranteed reachability analysis of stochastic hybrid systems," in *Proc. of SNR'15*, ser. EPiC Series in Computing, vol. 37. EasyChair, 2015, pp. 40–48.

[56] A. J. Thorpe and M. M. K. Oishi, "Model-free stochastic reachability using kernel distribution embeddings," *IEEE Control Systems Letters*, vol. 4, no. 2, pp. 512–517, 2020.

[57] G. M. Ziegler, *Lectures on Polytopes*, ser. Graduate Texts in Mathematics. Springer, 1995, vol. 152.

[58] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *Proceedings of the 8th International Workshop on Hybrid Systems: Computation and Control (HSCC'05)*, ser. LNCS, vol. 3414. Springer, 2005, pp. 291–305.

[59] M. Althoff, O. Stursberg, and M. Buss, "Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes," *Nonlinear Analysis: Hybrid Systems*, vol. 4, no. 2, pp. 233–249, 2010.

[60] X. Chen and S. Sankaranarayanan, "Model-predictive real-time monitoring of linear systems," in *Proc. of RTSS'17*. IEEE Press, 2017, pp. 297–306.

[61] M. Althoff, O. Stursberg, and M. Buss, "Safety assessment for stochastic linear systems using enclosing hulls of probability density functions," in *2009 European Control Conference (ECC)*, 2009, pp. 625–630.

[62] A. Girard and C. Le Guernic, "Zonotope/hyperplane intersection for hybrid systems reachability analysis," in *Proc. of HSCC'08*, ser. LNCS, vol. 4981. Springer, 2008, pp. 215–228.

[63] L. F. Cómbita, A. Cárdenas, and N. Quijano, "Mitigating sensor attacks against industrial control systems," *IEEE Access*, vol. 7, pp. 92 444–92 455, 2019.

[64] Y. Wang, Q. Liu, E. Mihankhah, C. Lv, and D. Wang, "Detection and isolation of sensor attacks for autonomous vehicles: Framework, algorithms, and validation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 8247–8259, 2022.

[65] L. Zhang, M. Liu, and F. Kong, "Demo: Simulation and security toolbox for cyber-physical systems," in *2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, ser. RTAS '23. Los Alamitos, CA, USA: IEEE Computer Society, May 2023, pp. 357–358. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/RTAS58335.2023.00040

[66] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A review of physics simulators for robotic applications," *IEEE Access*, vol. 9, pp. 51 416–51 431, 2021.

[67] K. Gamagedara, M. Bisheban, E. Kaufman, and T. Lee, "Geometric controls of a quadrotor uav with decoupled yaw control," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 3285–3290.

[68] T. Lee, M. Leok, and N. H. McClamroch, "Control of complex maneuvers for a quadrotor uav using geometric methods on se (3)," *arXiv preprint arXiv:1003.2005*, 2010.

[69] N. A. Chaturvedi, A. K. Sanyal, and N. H. McClamroch, "Rigid-body attitude control," *IEEE control systems magazine*, vol. 31, no. 3, pp. 30–51, 2011.

[70] J. M. Snider *et al.*, "Automatic steering methods for autonomous automobile path tracking," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009.

[71] Y. Tu, Z. Lin, I. Lee, and X. Hei, "Injected and delivered: Fabricating implicit control over actuation systems by spoofing inertial sensors," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1545–1562.

[72] MPC-Berkeley, "Berkeley autonomous race car (barc) repo," 2020. [Online]. Available: https://github.com/MPC-Berkeley/barc.git

[73] G. Huang and S. Lee, "Pc-based pid speed control in dc motor," in *2008 International Conference on Audio, Language and Image Processing*. IEEE, 2008, pp. 400–407.

[74] B. Messner, D. Tilbury, R. Hill, and J. Taylor, "Control tutorials for matlab and simulink: Aircraft pitch," *Re-trieved from https://web. archive. org/web/20200509164711/http://ctms. engin. umich. edu/CTMS/index. php*, 2020.

[75] N. Popovich, "Lateral motion control of boeing 747 by using full-order observer," in *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*, 2019, pp. 377–383.

[76] F. D. Sabatino, "Quadrotor control: modeling, nonlinearcontrol design, and simulation," 2015.

[77] S. N. Sheldon and C. Osmon, "Piloted simulation of an f-16 flight control system designed using quantitative feedback theory," *International Journal of Robust and Nonlinear Control*, vol. 9, no. 12, pp. 841–855, 1999.

[78] K. H. Johansson, "The quadruple-tank process: a multivariable laboratory process with an adjustable zero," *IEEE Trans. Control. Syst. Technol.*, vol. 8, pp. 456–465, 2000.

[79] M. Liu, L. Zhang, P. Lu, K. Sridhar, F. Kong, O. Sokolsky, and I. Lee, "Fail-safe: Securing cyber-physical systems against hidden sensor attacks," in *2022 IEEE Real-Time Systems Symposium (RTSS)*, ser. RTSS '22, 2022, pp. 240–252.